Distributed Neural Networks Microcontroller Implementation and Applications

Ioan Susnea

University “Dunarea de Jos” of Galati,
111, Domneasca Street,
Galati, 800201, Romania,
ioan.susnea@ugal.ro

Abstract: In this paper it is argued that, for any three-layer perceptron, it is always possible to design an equivalent distributed ANN, wherein the neurons are implemented on the nodes of a communication network, and the synapses between them are established in the communication process. In this approach, neurons are seen as processing and communication entities. Since both local and distributed implementations of a specific ANN are perfectly equivalent, they can use the same set of synapse weights, i.e. a distributed ANN can be trained on a local, equivalent software implementation. Two use cases are presented to demonstrate the validity of the idea.

Keywords: distributed ANN, microcontrollers, robot navigation, smart environment.

1. Introduction

Although artificial neural networks (ANNs) have been successfully used in almost any research field ([1], [2]), there are relatively few reports about microcontroller based implementations thereof ([3]).

While it is obvious that the severe limitations in what concerns hardware resources and computing power of the usual low cost microcontrollers make it difficult to use them for the implementation of ANNs, it is also clear that, in principle, it is possible to design systems wherein the computational task is divided in sub-tasks, executed by a plurality of microcontrollers, connected in a communication network ([4]).

Typical communication networks comprising nodes with limited data processing capabilities are the Wireless Sensor Networks (WSN – [5]). Several researchers noticed the similarities between WSN and ANNs, and proposed solutions for implementing neural network structures over WSN ([6], [7], [8]).

Another interesting research direction deriving from the concept of distributed neural network is the attempt to identify neural models of the interactions between agents in swarms ([9], [10).

In this paper, we go beyond the exploration of similarities and analogies between WSN and ANNs, and argue that, *for any three-layer perceptron, it is always possible to design an equivalent distributed ANN*. The neurons are implemented on the nodes of the communication network, and the synapses between them are defined in the process of communication.

Two use cases are presented to demonstrate the validity of the idea. In the first example, a mobile robot communicates with a plurality of “neural beacons”, or “neural landmarks”, and the resulting distributed ANN directly controls the navigation of the robot. In another example, a number of “smart” PIR (Passive Infrared) motion detectors, learn to control the HVAC system according to the activity patterns of the inhabitants, in order to reduce the overall energy requirements for heating the building.

Beyond this introduction, this paper is structured as follows:

Section II presents the principles of the implementation of distributed ANNs starting from the model of the three-layer perceptron.

Section III describes an example of using a distributed ANN, consisting in a number of “neural beacons” deployed in the environment, to control a mobile robot for path tracking.

Section IV presents a similar network, wherein the neurons are implemented by smart PIR sensors, used to adjust the reference temperature of a HVAC System in accordance with the occupancy patterns of the inhabitants.

Section V is reserved for conclusions.

2. Neurons as Communication Entities

A. Implementing detachable neurons

Consider a tree-layer perceptron as shown in Figure 1, containing generic neurons, like the one presented in Figure 2.

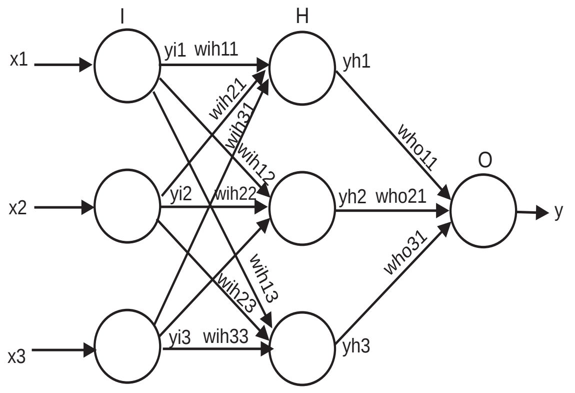


Figure 1. Typical three-layer perceptron

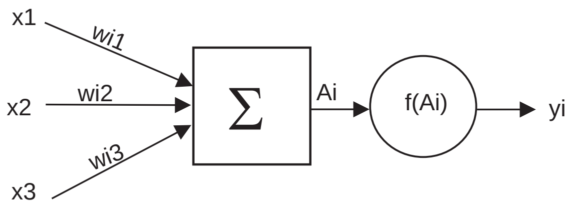


Figure 2. Structure of a generic artificial neuron

With the notations in Figure 2, A_i is the weighted sum of the inputs x_{ij}:

$$A_i = \sum_{j=1}^{N} w_{ij} x_j \qquad (1)$$

and the output $y_i = f(A_i)$ is, typically, a sigmoid function:

$$y_j = \frac{1}{1 + e^{-\sum_{j=1}^{N} w_{ij} x_j}} \qquad (2)$$

In terms of information processing, a neuron is a functional block capable to maintain a data structure like this:

```
struct neuron
{
 double x̄ [LAYER_SIZE];
 double w̄ [LAYER_SIZE];
 double A;
 double y;
}
```

where $\bar{x}$ is the vector containing the input values, $\bar{w}$ is a vector containing the weights of the incoming synapses (stored locally), A is computed locally according to (1), and y is the output value, computed locally with (2).

In a feedforward topology, the vector $\bar{x}$ contains the output values of the neurons of the previous layer. Obviously, these values can be transmitted either by direct connections between neurons, as depicted in Figure 1, or by means of messages, broadcasted over a communication network.

Assuming that:

– each neuron has an unique ID on the network,
– each neuron holds a list containing the IDs of the neurons with whom it makes synapses,
– each neuron stores the weights $\bar{w}$ of the incoming synapses,
– each neuron is capable to compute the values *A* and *y*, according to (1) and (2),
– each neuron broadcasts messages containing the current computed value of its output *y*,

then the ANN in Figure 1 can be implemented over a communication network, as shown in Figure 3.

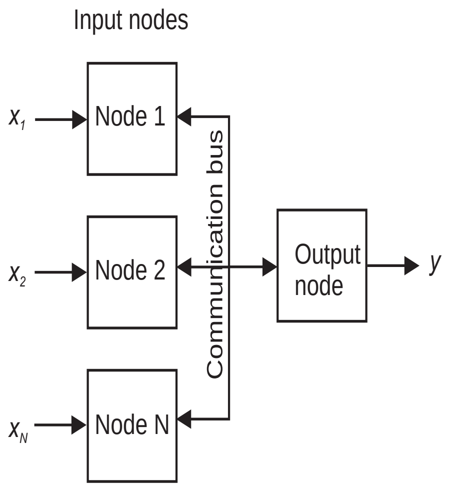


Figure 3. Sensor network compatible with distributed ANNs

In this parametrization, the neurons are entirely detachable, so that some or all of the neurons of the local implementation of the ANN can be implemented on a remote communication node.

Assuming that the communication network used is a sensor network that provides the input data for the whole ANN, and that the neurons on the input layer have linear transfer functions, the tasks associated with the network in figure 1 are distributed in the nodes of the communication network as shown in Figure 4.

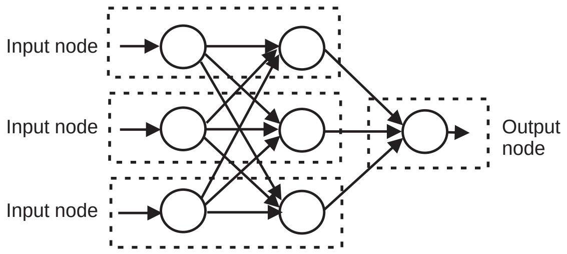


Figure 4. Distribution of tasks of the ANN over the nodes of the communication network

It is also important to note that the processing and communication tasks associated with one single neuron require relatively low computing power, and are compatible with almost any low cost microcontroller.

B. Training a distributed ANN

Since the distributed ANN (DANN) built this way is entirely equivalent with the perceptron model, it means that both distributed and non-distributed implementations of this model can share the same set of synapse weights, obtained in a process of training. So, the DANN can simply borrow the knowledge acquired by an equivalent non-distributed implementation using a recorded dataset. This can be done by broadcasting the weights of the synapses over the communication network, and instructing the detached neurons to identify and store the weights of their own incoming synapses.

It is also relatively easy to implement the backpropagation learning algorithm ([11]) on a DANN.

According to this algorithm, the approximation error of the network, defined by (3)

$$E(\bar{x},\bar{w},\bar{d}) = \sum_j (y_j(\bar{x},\bar{w}) - d_j)^2 \qquad (3)$$

propagates backwards from neuron j to neuron i, so that the error of the neuron i (δE_i) is a fraction of the general error, proportional with the weight of the synapse between neurons i and j (4).

$$\delta E_i = w_{ij}\delta E_j \qquad (4)$$

The network "learns" by iteratively adjusting the weights of the synapses so that the gradient of the error descents:

$$w'_{ij} = w_{ij} + \Delta w_{ij} = w_{ij} - \eta \frac{\partial E}{\partial w_{ij}} \qquad (5)$$

To implement the backpropagation algorithm on a DANN, we need to extend the data structure associated with the neurons as follows:

```
struct neuron
{
 double x̄ [LAYER_SIZE];
 double w̄ [LAYER_SIZE];
 double A;
 double y;
double delta;
double new_ w̄ [LAYER_SIZE];
}
```

The variable *delta* is δE_i ,computed with (4) and broadcasted along with *y*, and the vector *new_* $\bar{w}$, computed locally with (5), by each neuron, contains the adjusted weights during training.

C. Related work

Perhaps the most typical example of communication protocol, wherein all messages are broadcast messages is CAN - Controller Area Network (12). CAN seems to be intrinsically suitable for DANN applications, but it has a major disadvantage: there are few known implementations of wireless CAN (13).

But even in wired networks, CAN is still a powerful tool, and the most significant works describing neural structures over communication networks ([14],[15]) are based on CAN.

3. Neural Beacons for Robot Navigation

A. Description of the experiment

Consider the experimental setup shown in figure 5. In the first stage of the experiment, a differential drive mobile robot R (Pioneer3-DX from MobileRobots [16]) is manually guided along a path in an environment containing a number of "beacons" (1-8), each having processing and wireless communication means. Beacons are located at known positions in the environment (see also Figure 7).

The robot carries its own localization system (odometry) and periodically sends data packets containing information about the current position, and the values of the speeds of the driving wheels (v_L, v_R). This data is recorded by a computer connected to the robot through a wireless link.

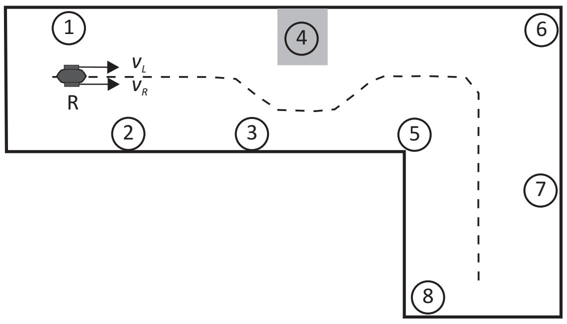


Figure 5. Mobile robot and beacons

In the second stage of the experiment, a dedicated software application implements the ANN in Figure 6, which is trained using backpropagation to approximate the functions (6) using data recorded in the first stage.

$$
\begin{aligned}
v_L &= f(t, d_1, d_2, d_3, ..d_8) \\
v_R &= f(t, d_1, d_2, d_3, ..d_8)
\end{aligned}
\tag{6}
$$

where $d_1,..,d_8$ are the distances between the robot and the beacons 1,...,8, at the moment *t*.

Finally, the microcontroller units (MCU) located on the beacons, and another MCU carried by the robot are programmed to implement a distributed ANN, according to the principles described in Section II. The weights of the synapses obtained by training in the second stage were transferred in a nonvolatile EEPROM memory of the MCUs. The communication network is shown in Figure 7.

The microcontroller unit on the robot implements the two neurons of the output layer, and sends the values of $v_{L,}$, v_R to the robot via a second RS232 communication line.

The trajectory of the robot, under the control of the distributed ANN, recorded with the simulator MobileSim ([16]) is shown in Figure 8.

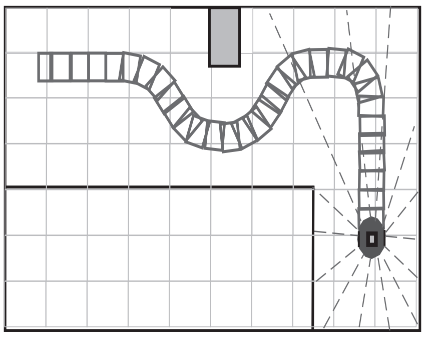


Figure 8. The trajectory of the robot recorded with MobileSim

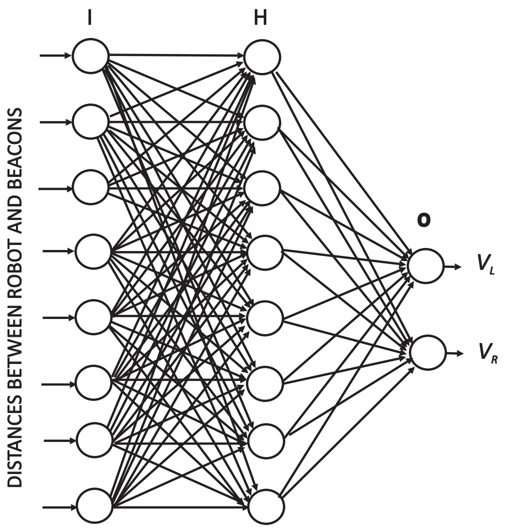


Figure 6. The ANN used to approximate v_L, v_R

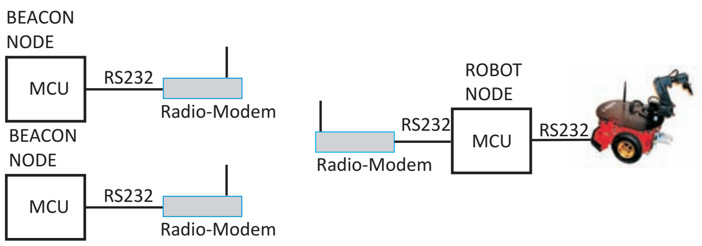


Figure 7. Robot and beacons in communication

A MATLAB simulation based on the same data set produced the trajectory depicted in Figure 9.

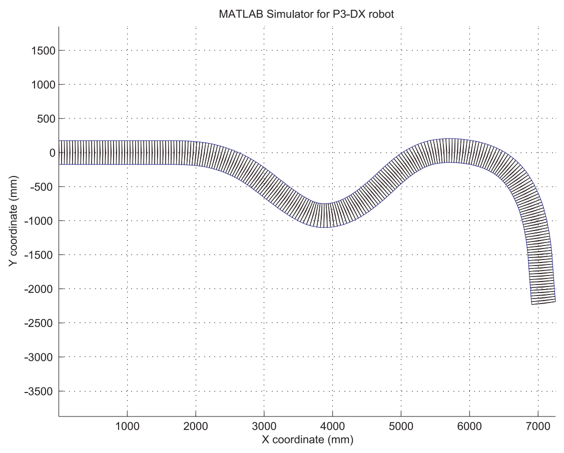
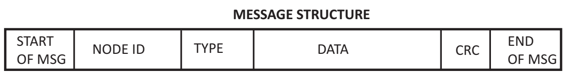


Figure 9. MATLAB simulation of the robot motion under the control of the ANN

B. Notes on the communication protocol

For simplicity reasons, we have used a specially designed MASTER-SLAVE protocol. The general structure of the messages is shown in figure 10.



MESSAGE STRUCTURE

START OF MSG	NODE ID	TYPE	DATA	CRC	END OF MSG

Figure 10. Structure of the messages over the communication network

Every 100ms, the MASTER broadcasts a Type 0 message, having the current coordinates of the robot in the DATA field. All the SLAVE nodes store the coordinates of the robot and each of the beacon nodes computes the relative distance between the robot and the corresponding beacon.

Next, the MASTER starts sending short queries having the SLAVE ID in the Type field, and 0 bytes in the DATA field. The SLAVE that recognizes its ID in the query, responds by broadcasting a message containing the computed value of the output y of its associated neuron. At the end of the cycle of queries-answers, the MASTER has all the required information to compute v_L and v_R ,and sends these values to the robot.

C. Related work

The experiment described in this section is, to some extent, related with those presented in [17], and [18]. In [17], Miglino et. al. use sonar readings as inputs for a microcontroller implementation of a local ANN that generates references for the speeds of the drive wheels of the robot.

In [18], Hellström and Ringdahl present the "follow the past algorithm" for path tracking – a method that also uses the past "experience" of the robot, but the similarities stop here.

4. Neural Control of the HVAC System in Smart Buildings

A. The idea

According to official estimates ([19]) buildings are responsible for 40% of the energy consumption in Europe. Under these circumstances, finding solutions to reduce energy waste in buildings may have significant impact on the overall energy consumption. For example, most modern HVAC systems (Heating Ventilation and Air Conditioning) can be programmed to automatically adjust the target temperatures depending on the time of the day, assuming that during certain time intervals, the building is likely to be unoccupied, and can be maintained at a lower level of thermal comfort. However, a priori predictions about building occupancy are seldom accurate; therefore this control scheme has limited efficiency.

The solution described here is based on the idea that the occupancy and activity levels can be estimated using passive infrared (PIR) sensor, usually present in most buildings as part of the security system. By adding "detachable neurons" implemented according to the principles presented in Section II to each sensor, and providing a means for the resulting network to directly interface the HVAC system, we get a structure like the one presented in Figure 11.

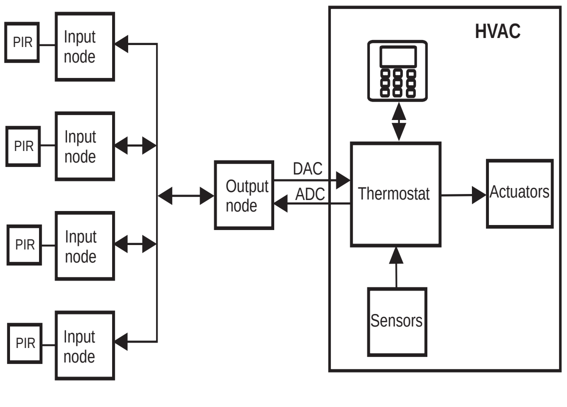


Figure 11. Controlling a HVAC with a distributed ANN

In this structure, it is assumed that the HVAC system accepts temperature references in the form of analogue signals, which can be generated by the output node of the distributed ANN using a DAC, and also can be read by an ADC during the training phase.

The input of the distributed ANN is a vector $\overline{x}$ of "occupancy quotients", computed by each input node by counting the digital pulses generated by the PIR sensors within a specified time frame, and the output of the network is the temperature reference for the HVAC.

In normal operation mode, the input nodes broadcast two types of messages:

– Type 1 messages, containing the components of the input vector $\overline{x}$. These messages are addressed to the other input nodes.
– Type 2 messages contain information about the computed components of the output vector $\overline{y}$ of the respective neuron. These messages are addressed to the output node.

Communication can be wired or wireless.

The activity of an input node is synthetically presented in Figure 12.

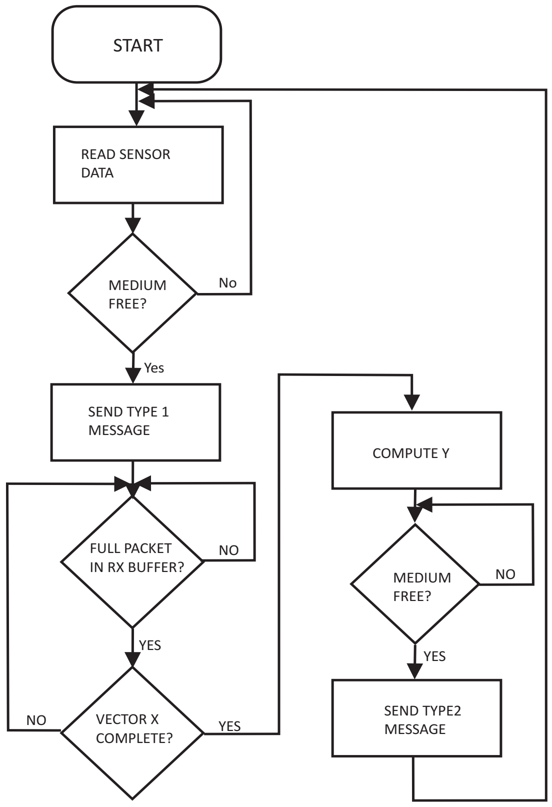


Figure 12. Normal operation diagram for an input node

In normal operation mode, the output node does not need to send messages over the communication medium. Its operation is described in Figure 13.

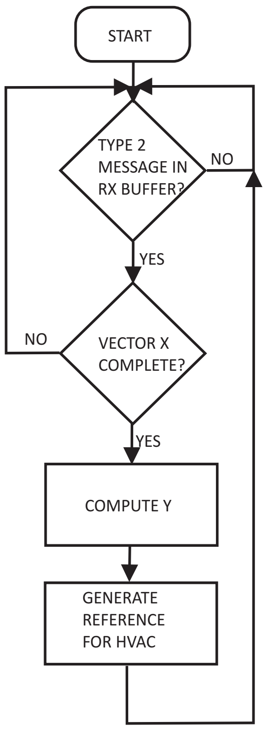


Figure 13. Normal operation diagram for the output node

During the training phase, the output node reads the "right" values of the reference temperature (manually adjusted), computes the approximation error and broadcasts "delta" messages over the communication network to implement the backpropagation algorithm, as described in section 2.

B. Related work

The idea of using smart sensors and neural networks to control HVAC systems is not new. For example, [20] describes a complex multi-purpose smart sensor incorporating a smoke detector, a PIR motion detector, and a light sensor, designed to control the lighting systems in buildings.

The patent application [21] describes a solution based on interfacing the security system with the HVAC in order to extract occupancy information by analyzing the states of the alarm system.

5. Conclusions

A method for implementing distributed ANNs according to the model of three layer perceptron was presented.

In this approach, neurons are seen as detachable processing and communication entities, requiring relatively low computing power, which makes them compatible with any low- cost microcontroller.

Two use cases presenting applications of the resulting networks were discussed.

The principles of the implementation of distributed ANNs presented here can serve as starting point for further research for creating neural structures on systems that are intrinsically distributed, like sensor networks, swarms, the Internet, etc. The possible applications are almost unlimited.

REFERENCES

1. WIDROW, B., D. E. RUMELHART MICHAEL, A. LEHR, **Neural Networks: Applications in Industry, Business and Science,** Communications of the ACM, Volume 37 Issue 3, March 1994.
2. HAGAN, M., H. DEMUTH, **Neural Networks for Control,** Invited Tutorial, 1999 American Control Conference, San Diego, June, 1999, pp. 1642-1656.
3. BINFET, J., B. M. WILAMOWSKI, **Microprocessor Implementation of Fuzzy Systems and Neural Networks,** in Neural Networks, 2001. Proceedings. IJCNN '01. International Joint Conference on, July 2001, pp. 234-239.
4. SOARES, S. G. et. al., **Building Distributed Soft Sensors**, International Journal of Computer Information Systems and Industrial Management Applications ISSN 2150-7988 vol. 3, 2011, pp. 202-209.
5. VIEIRA, M. A. M., C. N. COELHO, Jr., D. C. da SILVA, Jr., J. M. da MATA, **Survey on Wireless Sensor Network Devices**, Emerging Technologies and Factory Automation, 2003. Proceedings. ETFA '03. IEEE Conference Sept. 2003 pp. 537-544.
6. OLDEWURTEL, F., P. MAHONEN, **Neural Wireless Sensor Networks**, Systems and Networks Communications, 2006. ICSNC '06. International Conference on, pp.28-36.
7. ENAMI, N., R. A. MOGHADAM, K. DADASHTABAR, **Neural Network Based Energy Efficiency in Wireless Sensor Networks: A Survey**, International Journal of Computer Science & Engineering Survey (IJCSES) Vol.1, No.1, August 2010.
8. SHAH, K., M. KUMAR, S. INC, T. ADDISON, **Resource Management in Wireless Sensor Networks using Collective Intelligence**, International Conference on Intelligent Sensors, Sensor Networks and Information Processing, 2008. ISSNIP 2008, pp. 423–428.
9. PARUNAK, H. v. D., S. BRUECKNER, **The Cognitive Aptitude of Swarming Agents**, Vector Research Center, Ann Arbor, MI, 2009. www. newvectors.net/staff/parunakv/CASA.pdf
10. SUSNEA, I., G. VASILIU, A. FILIPESCU A. RADASCHIN, **Virtual Pheromones for Real-Time Control of Autonomous Mobile Robots**, in Studies of Informatics and Control, Vol 18, issue 3, 2009, pp. 233-240
11. RUMELHART D. E., G. E. HINTON, R. J. WILLIAMS, **Learning Representations by Back-propagating Errors**, Nature 323, October 1986, pp. 533-536.
12. BOSCH, R., **CAN Specification Version 2.0**, Robert Bosch GmbH, Stuttgart, 1991.
13. K.-R. SOHN, H.-J. LEE, Y. KIM, **Wireless CAN Communications based on White LED,** Third International Conference on Ubiquitous and Future Networks (ICUFN), 15-17 June 2011, pp. 127–130.
14. BONASTRE, A., R. ORS, J.V. CAPELLA, J. HERRERO, **Distribution of Neural-based Discrete Control Algorithms Applied to Home Automation with CAN,** Proceedings of the 8th International CAN Conference, Las Vegas, USA, February 2002.

15. CAPELLA, J. V., A. BONASTRE, R. ORS, **An Advanced and Distributed Control Architecture Based on Intelligent Agents and Neural Networks**, IEEE International Workshop on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications 8-10 September 2003, Lvov, Ukraine.

16. www.mobilerobots.com.

17. MIGLINO, O., H. LUND, S. NOLFI, **Evolving Mobile Robot in Simulated and Real Environment,** Artificial Life, vol. 2, no. 4, 1995, pp. 417-434.

18. HELLSTRÖM, T., O. RINGDAHL, **Follow the Past: A Path-tracking Algorithm for Autonomous Vehicles**, International Journal of Vehicle Autonomous Systems, Vol. 4, Nos. 2-4, 2006, pp.216-224.

19. **Directive 2010/31/Eu of the European Parliament and of the Council of 19 May 2010 on the Energy Performance Of Buildings**, Official Journal of The European Union L153, Vol. 53, 18 June, 2010

20. US patent US2008291036A1

21. US patent US2009/266904A1